Cyber Security Detection with Machine Learning

Nguyen Le Harsh Tomar Victoria Brooks

***Abstract*—This document discusses an attempt to determine the appropriate supervised machine learning algorithms to detect malicious activities based on networking records.**

I. Introduction

This paper presents the project's plan, design, implementation, result/outcome. The UNSW-NB15 dataset is a network intrusion dataset. In our project, we looked at Decision Tree, Random Forest, XGBoost, and Support Vector Machine algorithms to look at the accuracy of the algorithms in detecting the different types of intrusions in the dataset.

II. Planning

When it comes to planning, there are a few major criteria that we needed to nail down in order to achieve our objectives in building a capable cyber security detector and accumulate valuable experience in solving a cyber security problem with machine learning. We decided that the most appropriate metrics to measure a capable machine learning model and its usefulness to our needs would be accuracy and performance. Therefore, we started to design a process to compare different machine learning algorithms based on said metrics. On the other hand, one of the factors that is not on our criteria list, but is important to our project, is our experience in the field of machine learning and data science. We have decided to focus on supervised learning approach. Moreover, one of the traits that would benefit us tremendously in the project is how much computational resources needed to run supervised algorithms. It is much more appropriate to the resources that we were able to acquire. Within the supervised machine learning space, we discovered that Decision Tree and Random Forest were the best fit for our project because of a few reasons. Firstly, Decision Tree allows us to fully analyze the possible consequences of a decision with robust visualization. This is a tremendous advantage for an inexperienced team like us. Secondly, it also provides a framework to quantify the values of outcomes and probabilities of achieving them. Lastly, Random Forest is essentially built on top of Decision Tree. That will help us ease our learning journey into machine learning. Furthermore, having the desire to learn more, we would also experiment with the Gradient Boosting and Support Vector Machine (SVM) algorithms. Gradient Boosting is also an algorithm that is derived from Decision Tree; therefore, it will not only provide us with an opportunity to diverse our project, but it will also render our transition into a new algorithm with less friction. On the other hand, SVM will provide a different look into supervised machine learning considering it is not in the family with Decision Trees like the others that we planned to use in the project.

Into the next part of planning phase, we needed to figure out how to distribute our human resources to achieve the optimal results for project. Which means achieving project objectives as well as fulfilling our desire to learn more about machine learning. We have decided to evaluate the strengths, weaknesses, and interests of each team member. We also utilized the application workflow of Data Science in Cyber Security provided in class to guide our planning, design and implementation phases. By closely following the workflow, we were able to draw out the steps of projects quite effectively and thoroughly in the design phase.

III. Design

In the design phase of the project, we spent most of our time and efforts on figuring out how to carry out our project most effectively. There are many different aspects that we had to pay appropriate attention to such as technology, methodologies, and approaches.

As novices in the field of machine learning, selecting the suitable set of technologies were vital to our project. We decided to go with the more popular choices of libraries. The reason behind our decision was that there would be more available support and guidance that could help us overcome certain technical challenges. Moreover, their documentation is typically extensive, which will aid us significantly in our applications. This is the main reason why we went with scikit-learn, pandas and NumPy for almost all operations in the project. One of the main advantages of these three popular libraries is that they were pre-built with all the functionalities that we needed. From data cleaning, data manipulation, training and testing machine learning models, to data visualization and plotting.

As mentioned earlier, we followed closely the workflow of Data Science in Cyber Security to determine how to achieve the objectives of the project. Going through the first phase, which is to import data, we needed to figure out where we can acquire the UNSW datasets as well as how we can all work on the datasets in a uniform matter. In the next phase, exploratory data analysis, our team spent time learning about the datasets itself. By having a proper understanding of the features and the different attacks included, we were in a much better place carrying out the later phases. This also tremendously assisted us in the phase after, data preprocessing and cleansing. In this phase, we had to particularly extend additional effort to research about feature selection techniques. We have decided to use various techniques that has been built on top of the algorithms that is used throughout our project such as Random Forest and Gradient Boosting. For Random Forest, we can utilize the Mean Decrease Impurity to rank the importance of each feature to the model and the Permutation Importance technique to achieve the same objective. On the other hand, in Gradient Boosting, we can also utilize the construction of the boosted decision trees model to plot the importance of all features towards the accuracy of the model. We planned to go through this process multiple times to compare the changes in accuracy to determine which set of features will be optimal. Also, through our findings on various published research on the same dataset, it is difficult to achieve high accuracy across all ten categories of traffics. Therefore, we also constructed additional steps improve the three attack categories with the least accuracy. In the training and validation phases, we planned split the datasets into an optimal training and testing portions. Which will ensure the

model having adequate data for training and validating. Through these two phases, we would go back and forth with different set of features discovered through the pre-processing phase and compare the results.

IV. Implementation

Data Cleaning & Preprocessing

In the data cleaning phase, initially we had to skim through the datasets to ensure that we had a strong grasp of it. By having gone through it, we were able to identify formatting issues needed to be fixed in order to optimize the training model later. Moreover, we also converted all the categorical values into numeric value to boost the performance of the algorithms in training phase. We performed conversion in all columns with non-numeric values. Later on, we also utilize the concept of One-Hot encoding on the attack categories column when we trained models for specific attacks. This allowed us to quickly create a label column for the specific attacks that we wanted to train the supervised models on.

In the preprocessing phase, we particularly paid extensive attention to the feature selection processes. Because it could determine the success of our project. Firstly, we trained a model using each of the supervised algorithms that we planned to use for the entire project. For this step, we trained with all the provided features to essentially use the results as benchmarks for later training and validation. Afterward, we started to apply the three feature importance techniques: Random Forest's feature importance based on mean decrease impurity, permutation, and Gradient Boosting's plot feature importance. After getting the sets of ranking of which features are important to the model, we then train model based on the top ten and top five important features. In this step, we would re-apply all selected supervised algorithms, so that we can compare with benchmarks and determine whether these features improve the models. By thoroughly keep track of the results on each model using each set of features, we would be able to compare results and reach our conclusions for the project.

Decision Tree

We started off our training phase with Decision Tree, as it is considered the most user-friendly algorithm. As previously designed, we initially trained the Decision Tree with all the provided features as benchmarks for later experimentations.

Then we started to apply the three selected feature selection techniques: mean decrease impurity, permutation feature importance and plot feature importance. In this step, we evaluated and ranked the importance of all features. After this step, we would start train different algorithms with the top 10 and top 5 important features based on the ranking from each technique. By doing this, we will be not able to validate whether a selected set of features is better than using the entire set. But we will also be able to evaluate which technique is most optimal for the datasets based on the improvement or changes in accuracy.

Random Forest

Random Forest Classification is a way to use decision trees in regression and in classification. We used Random Forest Classifier from sklearn. We trained the model and kept track of the f-scores and the accuracies in the different scenarios.

Gradient Boosting

For Gradient Boosting, we used XGBoost as the algorithm. XGBoost is a gradient boosting decision tree algorithm. We ran this algorithm on the data and looked at the accuracies for each of the attacks.

Support Vector Machine (SVM)

This algorithm is different from decision tree algorithm as if we compare both on basis of accuracy SVM is more accurate then DT. SVM uses kernel trick to solve non-linear problems whereas DT derive hyper-rectangles in input space to solve the problem.

During the SVM implementation, we have trained the model with all the training sets and predicted the result. Because it was taking too long to predict the whole data set, we have trained and predicted the first 2000 and 5000 lines of the dataset. We tried to train the whole dataset but because of not having much computing power in the laptop it took 2 days without showing any sign of result. Therefore, we went for 2000-5000 lines of dataset for the training.

To show data in the graphical form, we have also plotted the classifier of our program using any 2 datasets as X, Y coordinate.

Individual attack categories

If we compare the accuracy of all the attacks, we have seen that DOS, analysis, and backdoors attacks have the least accuracy when compared to others. Therefore, to further analyze them we have taken them individually and trained them for 7 features and predict their accuracy. As per the results, we were able to achieve good accuracy for each individual attack on 7 features. We have used a Decision tree and a Random Forest algorithm to train them and predict them.

V. Results

Decision Tree

As expected, our benchmark for Decision Tree algorithm was not that high. This is foreseen, because the model is being trained on ten different categories. However, there are a few categories that were able to get high performance such as Reconnaissance with F-1 score at 0.82. Therefore, we believed that if we would be able to improve the model performance with feature selection techniques.

Feature Selection

Due to the differences in approach, the rankings did not come back identical for all three techniques. However, they shared a large number of similarities especially the appearances of a few features as well as their ranks. We can see this trend in the figure 1, 2, and 3. With the rankings on hands, we will start training model using the top 10 and top 5 features selected from each technique and compare the results. We found out that each set of features affect the performance of the decision tree models differently. It appears that the Mean Decrease Impurity technique improve

the F-1 score on detecting the Worms and DoS categories. On the other hand, Gradient Boosting does not appear to improve much on the model. Lastly, Permutation technique improved the performance on the models on multiple categories. This technique deemed to be most effective in selecting the features that the model depends on. The technique is defined to be the decrease in model score when a single feature is randomly shuffled. And this process breaks the relationship between the feature and the target, hence, the decrease in score will indicate how much the model depends on the said feature.

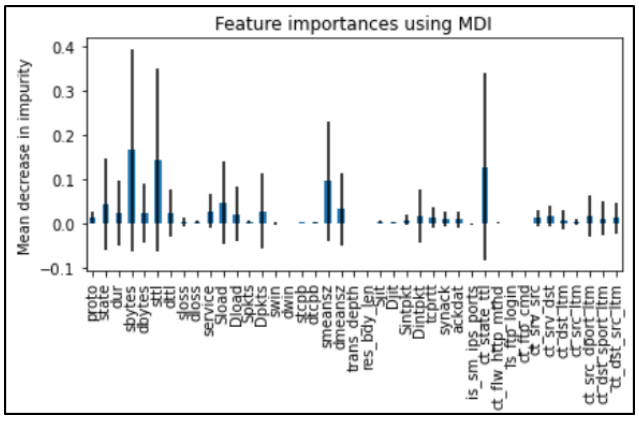

Figure 1. Feature ranking using Mean decrease in impurity technique

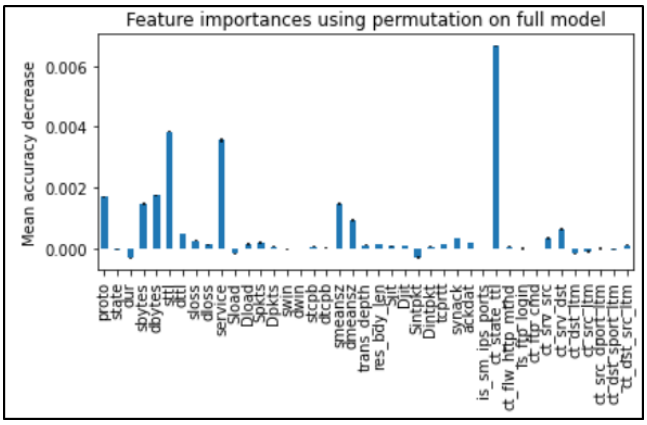

Figure 2. Feature ranking using permutation technique on full model

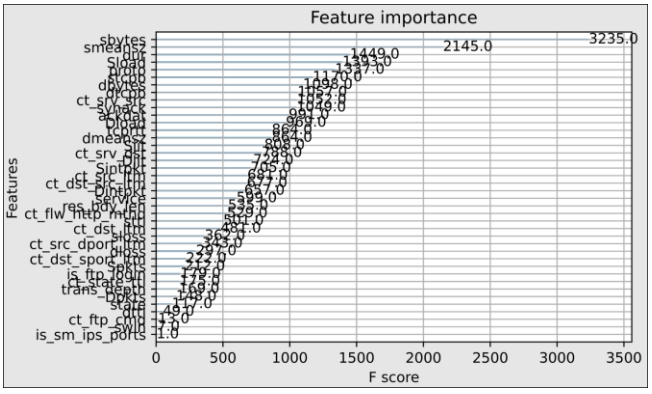

Figure 3. Feature ranking using Gradient Boosting

The top 10 set of features from this technique improves performance on all categories. Some drastically, some minimally. But they were all affected positively. However, the top 5 set of features does not appear to be that effective. Conclusively, we can see clearly that the permutation technique is most appropriate for the project. Moreover, the number of features should at a balanced point. Too little features used to train the model can result in poor performance just as much as too many features.

Random Forest

The Random Forest Classifier returned overall high accuracy when looking at all attacks, but specifically suffered in predicting Backdoors, Worms, and Analysis attacks. This is shown through the relatively low f1-scores given when looking at the classification report given when using all features. When the algorithm is trained on the top ten features, the Random Forest Classifier did not improve.

Random Forest — Scenario: All Features

	precision	recall	f1-score	support
Normal	1	1	1	665706
Reconnaissance	0.92	0.76	0.83	4208
Backdoors	0.51	0.09	0.15	658
DoS	0.34	0.27	0.3	4934
Exploits	0.63	0.81	0.71	13343
Analysis	0.63	0.08	0.13	820
Fuzzers	0.69	0.57	0.63	7370
Worms	0.44	0.16	0.24	49
Shellcode	0.66	0.61	0.63	457
Generic	1	0.99	0.99	64470
accuracy			0.98	762015
macro avg	0.68	0.53	0.56	762015
weighted avg	0.98	0.98	0.98	762015

Figure 4. Random Forest metrics with all features

Random Forest's Permutation Feature Importance — Scenario: top 10 features - *Features: 'ct_state_ttl', 'service', 'sttl', 'sbytes', 'dbytes', 'proto', 'smeansz', 'dmeansz', 'ct_srv_src', 'ct_srv_dst'

	precision	recall	f1-score	support
Normal	1	1	1	665706
Reconnaissance	0.91	0.76	0.83	4208
Backdoors	0.76	0.12	0.2	658
DoS	0.41	0.2	0.27	4934
Exploits	0.62	0.86	0.72	13343
Analysis	0.57	0.08	0.14	820
Fuzzers	0.66	0.57	0.61	7370
Worms	0.63	0.59	0.61	49
Shellcode	0.66	0.63	0.64	457
Generic	1	0.99	0.99	64470
accuracy			0.98	762015
macro avg	0.72	0.58	0.6	762015
weighted avg	0.98	0.98	0.98	762015

Figure 5. Random Forest with top 10 features

Gradient Boosting

The XGBoost algorithm used high accuracy, but not actual high accuracy when looking at the f1-scores for each attack. The XGBoost algorithm especially struggles with Backdoors, DoS, and Analysis attacks. However, the ability to classify the Exploits, Fuzzers, Worms, and Shellcode falls solidly in the “okay” territory. While not good, the f1-scores for each of these was also not bad, given that they are over 0.5.

Individual attack categories

Whereas if we compare the accuracy of the 3 individual attacks now, we have almost achieved 99%-100% results compared to 10%-15% initially.

DOS

Decision Tree

	precision	recall	f1-score	support
0	0.99	1.00	1.00	757081
1	0.28	0.07	0.11	4934
accuracy			0.99	762015
macro avg	0.64	0.53	0.55	762015
weighted avg	0.99	0.99	0.99	762015

Randon Forest

	precision	recall	f1-score	support
0	0.99	1.00	1.00	757081
1	0.44	0.04	0.07	4934
accuracy			0.99	762015
macro avg	0.72	0.52	0.53	762015
weighted avg	0.99	0.99	0.99	762015

Figure 7. Individual DOS Attack Results.

Analysis

Decision Tree

	precision	recall	f1-score	support
0	1.00	1.00	1.00	761195
1	0.35	0.09	0.14	820
accuracy			1.00	762015
macro avg	0.67	0.54	0.57	762015
weighted avg	1.00	1.00	1.00	762015

Random Forest

	precision	recall	f1-score	support
0	1.00	1.00	1.00	761195
1	0.72	0.08	0.15	820
accuracy			1.00	762015
macro avg	0.86	0.54	0.57	762015
weighted avg	1.00	1.00	1.00	762015

Figure 8. Individual Analysis Attack Results.

Backdoor Attack

Decision Tree

	precision	recall	f1-score	support
0	1.00	1.00	1.00	761357
1	0.40	0.07	0.11	658
accuracy			1.00	762015
macro avg	0.70	0.53	0.56	762015
weighted avg	1.00	1.00	1.00	762015

Randon Forest

	precision	recall	f1-score	support
0	1.00	1.00	1.00	761357
1	0.71	0.05	0.10	658
accuracy			1.00	762015
macro avg	0.85	0.53	0.55	762015
weighted avg	1.00	1.00	1.00	762015

Figure 9. Individual Backdoor Attack Results.

VI. Conclusions

Through this research, we were able to determine the best algorithm for this classification task. The best algorithm would be the Random Forest Classification algorithm due to its high accuracy and avoiding overfitting from using multiple decision trees. Also, we learned that the data preprocessing phase is just as important as the training and testing phase. Based on the results of our experiments, this is especially true for the decision tree, as decision trees are prone to overfitting from either too many features or too few features. Too few features also decrease the performance of the models, as shown by the models trained on the top 5 features against the top 10 features. The permutation feature selection technique was the best technique performed because it repeats calculations to remove bias to obtain the best score.

References

[1] T. Shin, “Understanding feature importance and how to implement it in Python,” Medium, 26-Feb-2021. [Online]. Available: https://towardsdatascience.com/understanding-feature-importance-and-how-to-implement-it-in-python-ff0287b20285. [Accessed: 29-Apr-2022].

[2] O. Sagi and L. Rokach, “Approximating XGBoost with an interpretable decision tree,” *Information Sciences*, vol. 572, pp. 522–542, 2021.

[3] Moustafa, Nour & Slay, Jill. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). 10.1109/MilCIS.2015.7348942.

[4] M. Rostami and M. Oussalah, “A novel explainable COVID-19 diagnosis method by integration of feature selection with random forest,” Inform. Med. Unlocked, vol. 30, no. 100941, p. 100941, 2022.